(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 810 756 A2

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
03.12.1997 Bulletin 1997/49

(51) Int Cl.[6]: **H04L 12/24**

(21) Application number: **97303168.5**

(22) Date of filing: **09.05.1997**

(84) Designated Contracting States:
DE FR GB NL SE

(30) Priority: **29.05.1996 US 654903**

(71) Applicant: **SUN MICROSYSTEMS, INC.**
Mountain View, CA 94043 (US)

(72) Inventors:
- **Yamunachari,Sundararajan**
  Sunnyvale California 94086 (US)
- **Rangarajan ,Govindarajan**
  Sunnyvale California 94087 (US)

(74) Representative:
Cross, Rupert Edward Blount et al
BOULT WADE TENNANT,
27 Furnival Street
London EC4A 1PQ (GB)

(54) Customizable automatic management of network devices

(57) Automatic management of a plurality of network devices is performed by sending customized event requests to the devices. The customized event requests specify variables and thresholds of interest to the device being managed. The customized event requests can be generated and fired by a network manager. Predefined event requests are read from computer memory. Each predefined event request includes a variable bindings list having at least one variable/threshold pair. The variable in each pair corresponds to a variable whose value is stored in the Management Information Base of a respective network device. The threshold in each pair is a threshold value for the variable in that pair. Each variable/threshold pair defines an event corresponding to the variable in the pair reaching a value having a predefined relationship to the threshold in the pair. Network devices are associated with the predefined event requests and the requests are sent to their associated devices. Response messages are then received from the devices in accordance with the predefined event requests sent to them and the status of the responding devices is indicated.

EP 0 810 756 A2

Printed by Jouve, 75001 PARIS (FR)

Description

Background of the Invention

The present invention relates to digital communications. More specifically, the present invention relates to network management.

Today, large numbers of personal computers and workstations are being interconnected with file servers, print servers, modems, hubs and other devices to form local area networks, metropolitan area networks and wide area networks. These networks allow the personal computers and workstations to share information and valuable resources among each other. Now more than ever, individuals and companies depend on networks to conduct business and to communicate with people around the world. Indeed, the network has become the computer.

Most networks use a network manager and some form of Simple Network Management Protocol (SNMP) for managing the network. Among its management tasks, the network manager automatically monitors the status of the devices on the network. The network manager sends predefined event requests to the devices, which are requested to return responses when certain events occur. For example, a disk agent might be requested to send a response if available disk space falls below 50%.

There are times when an event request might not define important events for a device, especially when the device is manufactured by more than one vendor. An SNMP-manageable device stores in its memory a Management Information Base (MIB), a collection of objects or variables representing different aspects of the device (e.g., configuration, statistics, status, control). For each class of device, the MIB has a core of standard variables. Each vendor of a device will add to the core, variables that it feels are important to the management of its device. Thus, the MIB for a router from a first vendor might be different from the MIB for a router from a second vendor, and an event request that defines important events for the one router might not necessarily define the same important events for the other router. This is a problem with network managers.

Another problem is that current network managers do not allow the health of the devices to be defined. The health is "hard-wired" into the event requests. Yet the desired definition of a healthy device or system might not concur with the predefined definitions. For example, a router could have five devices attached, two of which are redundant. Even if one of the redundant devices is bad, the router is still good. However, the network manager would indicate that the router is bad.

Summary of the Invention

These problems may be overcome by methods, computer storage medium and apparatus according to the present invention. A method of monitoring a plurality of network devices comprises the steps of reading a plurality of predefined event requests from computer memory; associating at least some of the devices with the predefined event requests; and sending the predefined event requests to their associated devices.

A method of generating an event request for a network device comprises the steps of forming a plurality of predefined event requests; and associating one of the plurality of predefined event requests with the device. At least two of the plurality of predefined event requests are formed by different variable bindings lists.

A computer storage medium stores a plurality of executable instructions which instruct a computer to automatically monitor a plurality of network devices. The plurality of instructions comprises instructions which instruct the computer to read a plurality of predefined event requests; instructions which allow the computer to associate the predefined event requests with at least some of the network devices; and instructions which instruct the computer to send the predefined event requests to their associated devices.

Apparatus for automatically monitoring a plurality of devices on a network comprises memory for storing a plurality of different predefined event requests; and a processor for reading the predefined event requests, associating the event requests with at least some of the devices in response to user inputs, and firing the associated event requests.

A network manager for communicating with a plurality of network agents comprises a computer that is programmed to read a plurality of predefined event requests; associate the predefined event requests with the plurality of agents; and send the predefined event requests to their associated agents.

Brief Description of the Drawings

Figure 1 is a schematic diagram of an internet including a network manager embodying the present invention;

Figure 2 is a block diagram of the network manager embodying the present invention;

Figure 3 is a static object oriented model of software that is executed by the network manager embodying the present invention;

Figure 4 is an illustration of a dialog box for customizing event requests, the dialog box being generated by the network manager;

Figure 5 is a flowchart of initial steps performed by the network manager;

Figure 6 is a flowchart of steps for monitoring the health of the network;

Figure 7 is a flowchart of steps for creating customized requests; and

Figure 8 is a flowchart of steps for creating a predefined event request.

Detailed Description of the Invention

Figure 1 shows an internet 10 including first, second, third and fourth subnetworks S1, S2. S3 and S4 that are interconnected by intermediate systems 12, 14, 16 and 18. For illustrative purposes only, the first and second subnetworks S1 and S2 are of a first topology, such as Token ring, the third and fourth networks S3 and S4 are of a second topology, such as Ethernet, the first and second intermediate systems 12 and 14 are bridges 12 and 14, and the third and fourth intermediate systems 16 and 18 are first and second routers 16 and 18. The first router 16 is manufactured by one vendor and the second router 18 is manufactured by a different vendor. MIBs of the first and second routers 16 and 18 have the same core of variables, but they have different variables added to their cores. Thus, the MIBs of the first and second routers 16 and 18 are different.

The first subnetwork S1 includes a first group of addressable devices 20, the second subnetwork S2 includes a second group of addressable devices 22, the third subnetwork S3 indudes a third group of addressable devices 24, and the fourth subnetwork S4 includes a fourth group of addressable devices 25. The addressable devices 20, 22, 24 and 25 can be devices such as workstations, personal computers, printers and hubs. Each device 20, 22, 24 and 25 is associated with an MIB and an agent. The agent is a software program that may or may not be resident in the device. TCP/IP is used to regulate how data is packeted and transported between the devices 20, 22, 24 and 25. Each device 20, 22, 24 and 25 has a physical or medium access control (MAC) address and a unique IP address.

The internet 10 also includes a network manager 26, which is connected to the third subnetwork S3 and which has access to the MIB of each device 20, 22, 24 and 25. SNMP is the protocol used for managing the devices 20, 22, 24 and 25. The agents allow a network manager to access the MIB of each device. Such accessibility allows the network manager to control operations of the agents, analyze resource performance, identify and resolve faults, and automate management tasks. For example, the network manager can request an agent to change the value of a variable in the MIB of a device, and it can request an agent to send out a response when an event occurs, such as a MIB variable exceeding a threshold for a device. For a general description of network management, see W. Stallings, "Data and Computer Communications", MacMillan (4th ed. 1994) pp. 701-24, which is incorporated herein by reference.

Figure 2 shows the network manager 26 in greater detail. The network manager 26 includes a workstation 28 such as a SPARCstation™ or SPARCserver™. Both of these workstations use a RISC-based high-performance "SPARC" microprocessor 30. The SPARCstation™, SPARCserver™, and "SPARC" microprocessor are all commercially available from Sun Microsystems, Inc., the assignee of the present invention. The workstation 26 is configured with a color display monitor 32 and a CD ROM drive 34 for distribution media. It is also configured with volatile memory 36 (e.g., 32 Mbytes of DRAM) and non-volatile memory 38 (e.g., a 400 Mbyte hard drive).

Software for the network manager includes a "UNIX"-based operating system 40. Operating systems for the "SPARC" microprocessor include "SOLARIS" 2.4 or greater and "SOLARIS" 1.x or later. The "SOLARIS" operating systems are also commercially available from the assignee of the present invention. The operating system 40 is stored on a portable computer memory medium (e.g., a CD ROM) and loaded onto the non-volatile memory 38 from the CD ROM drive 34.

Additional software for the network manager 26 includes a network topology database 42 and a Console program 44 that automatically manages and displays the devices indicated by the network topology database 42. The Console program 44 can be stored on a portable computer medium and loaded onto the non-volatile memory 38 from the CD ROM drive 34. The network topology database 42 can be created dynamically by a discover tool, which is executed by the workstation 26. The network topology database 42 is also stored in the non-volatile memory 38.

Referring now to Figure 3, the network topology database 42 is a collection of structure or schema files and instance files that describe the internet 10. Four basic elements are components (e.g., printers, routers, workstations), views (collections of elements, including other views), buses (e.g., a Token Ring segment) and connections (e.g., an RS-232 link). Structure files for other elements can be added to the network topology database 42. Each structure file includes a number of records that describe the structure of a particular element. The instance files contain instances of structure files for the elements that have been discovered on the internet 10. For a description of network topology databases, see C. Malamud, "Analyzing Sun Networks", Van Nostrand Reinhold (1992) pp. 419-21, which is incorporated herein by reference.

The Console program 44 includes an object-oriented, graphical user interface (GUI). The GUI can be derived from OpenWindows™ 3.1 or later or any other library of classes for GUIs. Inputs 45 are supplied to the GUI via a mouse and a keyboard. When executed, the Console program 44 displays a menu bar that allows the various features described below to be selected.

The Console program 44 has an Auto Management feature 46 that monitors the health of the internet 10. When the Auto Management feature is selected, a "Properties" dialog box is displayed. The Properties dialog box offers the following options:

Automatic Management: Enable/Disable
Polling Interval: <value>
Management Behavior: Default/Custom

Automatic Management is performed when the Enable button is selected and a value for the polling interval is specified. Event requests are automatically started for the devices 20, 22, 24 and 25 in the network topology database 42. The event requests are repeated at every polling interval. For example, a polling interval of 600 would cause event requests to be sent every 600 seconds. When an event request is fired, it is sent to an agent 47 of the device at the destination address specified in the event request's outer message wrapper.

If the Default button for Management Behavior is also selected, one of the following default event requests is sent at every polling interval (with order preserved):

(1)SNMP Event Request(sysUpTime increased by less than <number>).
(2)Hostperf Event Request(upTime increased by <number>).
(3)ICMP echo Event Request(reachable equal to false).

What these default event requests determine is whether the devices 20, 22, 24 and 25 in the network topology database 42 are operative. If a device supports SNMP, the network manager 26 sends the SNMP Event Request to the device. If sysUpTime has increased by less than a number such as 1, the device returns an SNMP Response message. If the device does not support SNMP, but does support Hostperf, the network manager 26 sends the Hostperf Event Request to the device. If upTime has increased by a number such as 1, the device returns a Hostperf Response message. If the device does not support SNMP or Hostperf, it is sent an ICMP Event Request, which requests the device to send back a response indicating reachability.

If the Custom button for Management Behavior is selected instead of the Default button, the network manager 26 sends customized event requests to associated component types. The customized event requests are read from the network topology database 42.

The Console program 44 also includes a Request Management feature 48 that allows the customized event requests to be generated. When the Request Management feature 48 is selected from the menu bar, a Customize popup dialog box appears. The Customize popup dialog box shown in Figure 4 includes three columns: a Component column, a Predefined Request column and a Customized Automatic Management Requests column. The Component column lists the components (e.g., bridge, genhost, genws, hub, ipc, router1, router2), which are read from the network topology database 42. Note that router1 and router2 correspond to the first and second routers 16 and 18, which were manufactured by different vendors. The components that are displayed might not constitute all of the devices 20, 22, 24 and 25 on the internet 10, but only those devices in a particular cluster or view. The Predefined Request column lists the predefined requests (e.g., if_System_ Reboot, when_Disk_is_Full, when_Printer_Error, when_System_is_Not_Reachable, router1_test, router2_test), which are read from a Predefined Request file stored in the non-volatile memory 38. Note that the router1_test for the first router 16 is different than the router2_test for the second router 18.

Customized event requests are generated by associating the components in the Components column with the predefined event requests in the Predefined Request column. A component and a predefined event request could be associated, for example, by dragging a component from the Component column over to a predefined event request from the Predefined Requests column. The Customized Automatic Management Requests column lists the pairs of associated components/requests. For example, router1 from the Component column is dragged over to router1_test from the Predefined Request column, and the router1:router1_test is displayed in the Customized Automatic Management Requests column. Only one predefined request is associated with each component type. This is done to reduce traffic on the network. Besides, multiple requests for a device are not really needed since each predefined event request can define multiple events. The predefined event requests displayed in the Customized Automatic Management Requests column are stored in the network topology database 42 in the instance files of their associated component. It is once again noted that the Customize dialog box of Figure 4 is merely exemplary, and that the interface for associating the components and predefined event requests is left up to the designer of the Console program 44.

When the button for Default Auto Mgmt for Components w/o Customized Requests is checked (near the top of the dialog box), the default event requests are sent as described above to the devices that have not been associated with predefined event requests. That is, one of default requests (1), (2) and (3) is sent to each of the devices that have not been associated with predefined event requests. If the button is not checked, no event requests at all are sent to those devices.

When the Create Predefined button (at the bottom of the dialog box) is clicked on, a Create Request window 50 is displayed. The Create Request window 50 allows a predefined event request to be created. For a general illustration of an event request, see C. Malamud, "Analyzing Sun Networks", Van Nostrand Reinhold (1992) pp. 421-24. The Create Request window 50 displays certain fields of a predefined event request, such as name of the event request, a destination address, a protocol data unit (PDU) type, a request-id, and a variable bindings list. The PDU type and request-id are already filled in, having been determined by the management protocol. The destination address is filled in after the predefined event request is associated with a component. The Create Request window 50 allows the user to fill in a name of the predefined event request

(e.g., Disk_Available) and fill in the field for the variable binding list. For example, ten pairs of variables/thresholds are added to the variable bindings list. Five of these pairs define events based on disk availability. This allows the single Disk_Available event request to instruct the genws to retum Response messages when disk availability exceeds, 5%, 15%, 50%, 75% and 90%. Similarly, the router1_test event request could request the first router 16 to send a Response message when a redundant device goes down, when a non-redundant device goes down, and when more than two of its devices go down. The thresholds are selected solely at the discretion of the network administrator or other superuser. After the fields are filled out, a save button is clicked on and the predefined event request is added to the Predefined Request file. Even though different devices have different functionalities, the event requests can be configured for the device under consideration. Thus, the network administrator can define the health of the device.

The Console program 44 also has a Display feature 52 that allows the devices 20, 22, 24 and 25 in the network topology database 42 to be displayed. The devices 20, 22, 24 and 25 are represented by glyphs 54. Glyphs 54 have attributes such as color and brightness that provide additional information about the devices they represent.

The Display feature 52 can display different views of the devices 12-22 of the internet 10. The views can be arranged in different ways. Views of desired system resources can be displayed. For example, a single view of all managed routers on the network could be displayed, regardless of the actual location of the routers in the network hierarchy. Or, views of devices can be arranged in a hierarchy to depict various levels of the internet 10. The highest level of the hierarchy can be displayed as a single cloud, which represents the internet 10. The network administrator can "navigate" through the internet 10 by simply pointing a mouse and double clicking the glyph which refers to the view. Double-clicking the cloud glyph of the internet 10 would cause the Console program 44 to display the next level, which would include cloud glyphs for the subnetworks S1, S2. S3 and S4, glyphs of the intermediate systems 12-18, and glyphs of the physical links.

Certain colors can notify the network administrator of events relating to a device. Perhaps the event was a router that went down or a hard drive having less than 10% of disk space available. A glyph for any of these devices could be set to a color such as red. If available disk space increased to 50%, the glyph could be changed to an orange color. If the disk space availability increased to 75%, the color of the glyph could be changed to green.

Event requests can be sent manually to selected devices. By double-clicking the glyph of the selected device, the network manager 26 looks up the associated event request in the network topology database 42 and fires the event request to the selected device. If a view is double-clicked, the event requests are fired for all of the devices in the view.

Figures 5 to 8 show the operation of the network manager 26. Reference is made first to Figure 5. When the Console program 44 is executed (step 100), the network manager 26 generates the GUI (step 102). The GUI displays the glyphs 54 (step 104). The network manager 26 can color-code the glyphs in response to the responses from the devices. Using a mouse or keyboard, the user selects a particular view of the network (step 106). The GUI also displays the menu bar (step 108) which allows the Auto Management and Request Management features 46 and 48 to be selected.

Figure 6 shows the steps for monitoring the health of the network. The user selects the Auto Management feature 46 from the menu bar of the Console Program 44 (step 200)and the network manager 26 generates the Properties dialog box (step 202). The user enables Automatic Management (step 204), fills in a value for the polling interval (step 206) and selects a Management Behavior (step 208). If the user selects Default behavior (step 210), the network manager 26 sends the default event requests at every polling interval to the devices 20, 22, 24 and 25 indicated in the network topology database 42 (step 212). If the user selects Custom behavior, the network manager 26 reads the customized event requests from the network topology database 42 (step 214) and sends the customized event requests at every polling interval to the devices 20, 22, 24 and 25 indicated in the network topology database 42 (step 216).

Figure 7 shows the steps for creating customized requests. The user selects the Request Management feature 48 from the menu bar of the Console Program 44 (step 300). The network manager 26 reads the components from the network topology database 42 (step 302) and the predefined event requests from the Predefined Request file stored in the non-volatile memory 38 (step 304). Then the network manager 26 generates the Customize popup dialog box, which displays the columns of components and predefined event requests (step 306). The user creates a customized request by selecting a component and associating it with a predefined event request (step 308). The network manager 26 displays the associated pair, that is, the customized request in a separate column(step 310). The network manager 26 also stores the customized request in the network topology database 42 in the instance file of the associated component (step 312).

Figure 8 shows the steps for creating a predefined event request. The user clicks on the Create Predefined Request button from the Customize popup dialog box (step 400) and the network manager 26 generates the Create Request window 50 (step 402). The Create Request window 50 displays the fields of a predefined event request (step 404). The PDU type and request-id are automatically filled in by the network manager 26 (step 406). The name of the predefined event request

and the variable/threshold pair(s) for the variable bindings list are filled in by the user (step 408). After the fields are filled out, the user clicks on the Save button and the predefined event request is added to the Predefined Request file in non-volatile RAM 38 (step 410). The destination address for the predefined event request is filled in after the predefined event request is associated with a component.

Thus disclosed is a customized apparatus and method that provides flexibility in monitoring an internet, a subnetwork or even a particular device on a network. Events can be defined by the network administrator, and each request can be configured for a specific device.

It is understood that various changes and modifications may be made without departing from the spirit and scope of the invention. The invention is not limited to the internet configuration shown in Figure 1. Moreover, the invention is not limited to network managers including workstations having RISC processors that run "UNIX"-based operating systems. For example, the network manager can include a personal computer having an x86 or "PENTIUM" processor that runs a 32-bit "UNIX"-based operating system such as "SOLARIS" 2.4. The operating system does not even have to be "UNIX"-based.

The software for the network manager is not limited to the objects or the object-oriented design shown in Figure 3. The software can be developed according to any methodology and any programming language. The Console program 44 is not limited to a GUI for displaying the elements and generating and firing the event requests. Display, generation and firing can be performed directly by the Console program 44.

Claims

1. A method of monitoring a plurality of network devices, comprising the steps of:

   reading a plurality of predefined event requests from computer memory;
   each predefined event request including a variable bindings list having at least one variable/threshold pair where the variable in each pair corresponds to a variable whose value is stored in a respect network device's Management Information Base and the threshold in each pair is a threshold value for the variable in that pair;
   each variable/threshold pair defining an event corresponding to the variable in the pair reaching a value having a predefined relationship to the threshold in the pair;
   associating at least some of the devices with the predefined event requests;
   sending the predefined event requests to their associated devices;
   receiving response messages from the devices in accordance with the predefined event requests sent to the devices; and
   indicating status of the responding devices according to the response messages that are received.

2. The method of claim 1, further comprising the step of generating at least one of the predefined event requests by filling in a variable/threshold pair in a variable bindings list of the event request.

3. The method of claim 1, further comprising the step of generating at least one predefined event request by filling in a plurality of variable/threshold pairs in a variable bindings list of the event request, whereby multiple events for the associated device are defined.

4. The method of claim 1, wherein the step of associating is performed by displaying a list of the devices, displaying a list of the predefined event requests, selecting a device from the list of devices and a predefined event request from the list of predefined event requests to be associated with the selected device, wherein the selecting step is performed on or more times for one or more respective devices.

5. The method of claim 1, further comprising the steps of

   sending default event requests to the devices that have not been associated with the predefined event requests; and

6. The method of claim 1, further comprising the step of automatically sending the event requests at specified intervals.

7. The method of claim 1, further comprising the steps of:

   monitoring at least some of the devices;
   displaying the devices that are being monitored, the displayed devices being represented by glyphs; and
   color-coding the glyphs to indicate the status of the devices being monitored.

8. Apparatus for automatically monitoring a plurality of devices on a network, comprising:

   memory for storing a plurality of different predefined event requests; and
   each predefined event request including a variable bindings list having at least one variable/threshold pair where the variable in each pair corresponds to a variable whose value is stored in a respect network device's Management Information Base and the threshold in each pair

is a threshold value for the variable in that pair; each variable/threshold pair defining an event corresponding to the variable in the pair reaching a value having a predefined relationship to the threshold in the pair;
a processor for reading the predefined event requests from the memory, associating the event requests with at least some of the devices in response to user inputs, and transmitting the associated event requests to the respective devices.

9. The apparatus of claim 8, wherein the processor also generates the predefined event requests and stores the generated requests in the memory.

10. The apparatus of claim 9, wherein the processor generates each predefined event request by receiving at least one variable/threshold pair and writing each received pair to the variable bindings list in the predefined event request.

11. The apparatus of claim 9, further comprising a video monitor and at least one input device, wherein the processor generates a GUI for receiving the at least one variable/threshold pair and for allowing the event requests to be associated with the devices.

12. The apparatus of claim 11, wherein the processor selectively displays glyphs representing the devices on the video monitor, and wherein the processor color-codes the glyphs to indicate status of the displayed devices.

13. The apparatus of claim 9, wherein the processor programmed to generate a predefined request by:

displaying a field for a variable bindings list of the event request;
accepting an input for at least one variable/threshold pair: and
writing the inputted pair to the variable bindings list.

14. The apparatus of claim 9, wherein the processor is programmed to:

display glyphs representing the devices
receive responses from the devices; and
display status of the devices based on the received responses.

15. The apparatus of claim 14, wherein the processor is programmed to indicate status by color-coding the glyphs.

16. The apparatus of claim 9, wherein the processor is programmed to read, associate and send the predefined event requests when custom automatic management is selected, and wherein the processor is further programmed to send default event requests to the devices when default management is selected.

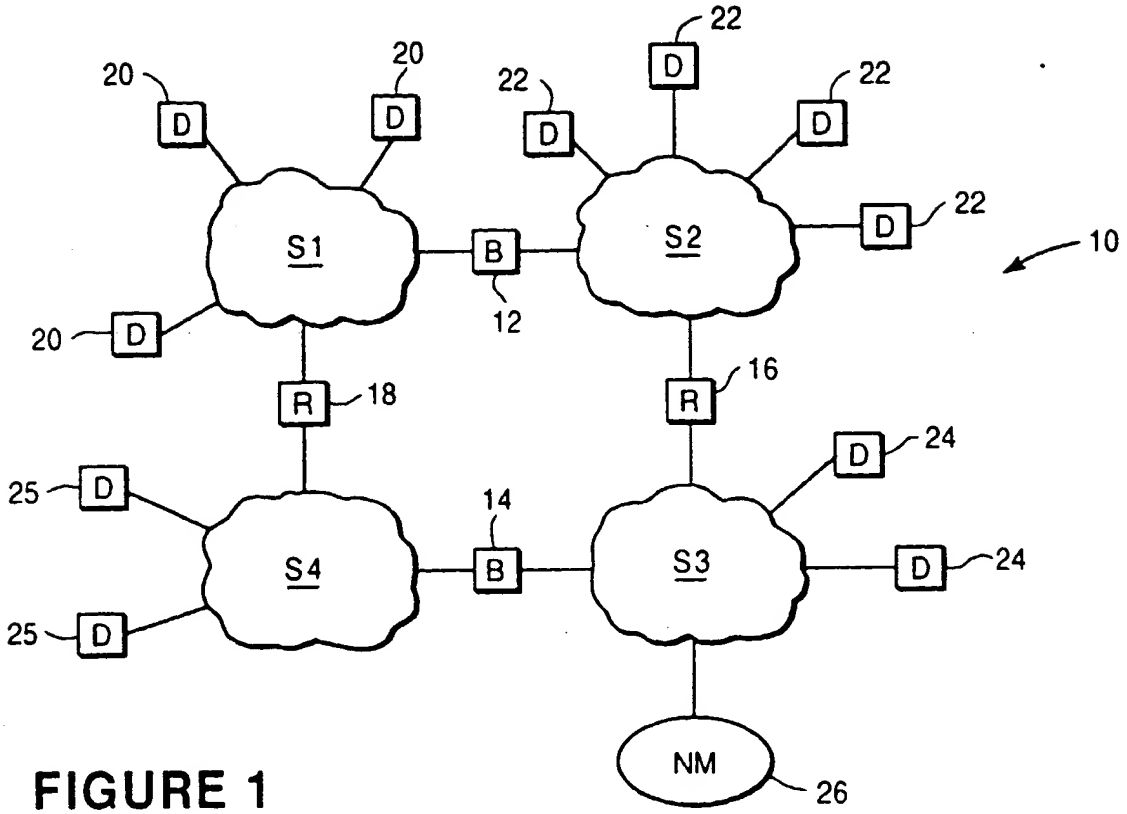


FIGURE 1

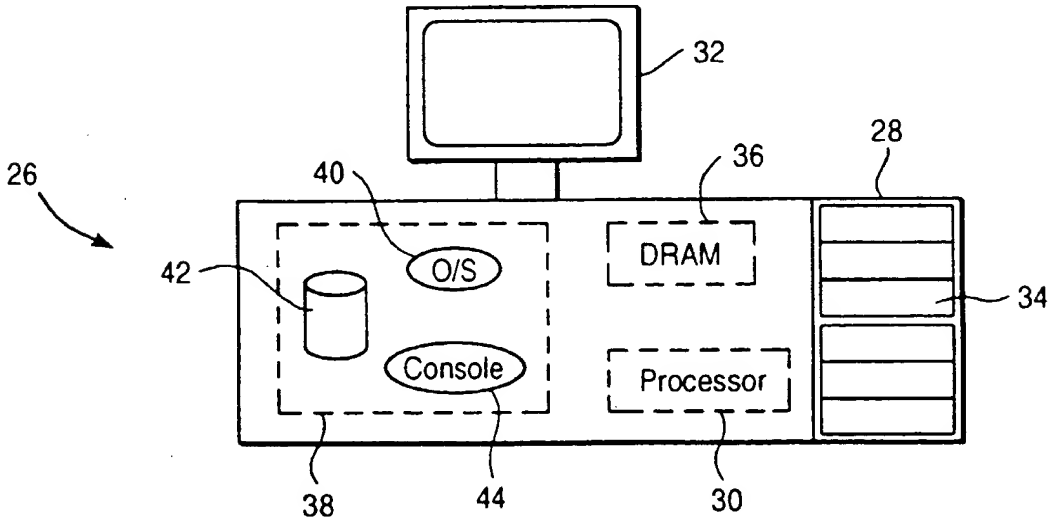


FIGURE 2

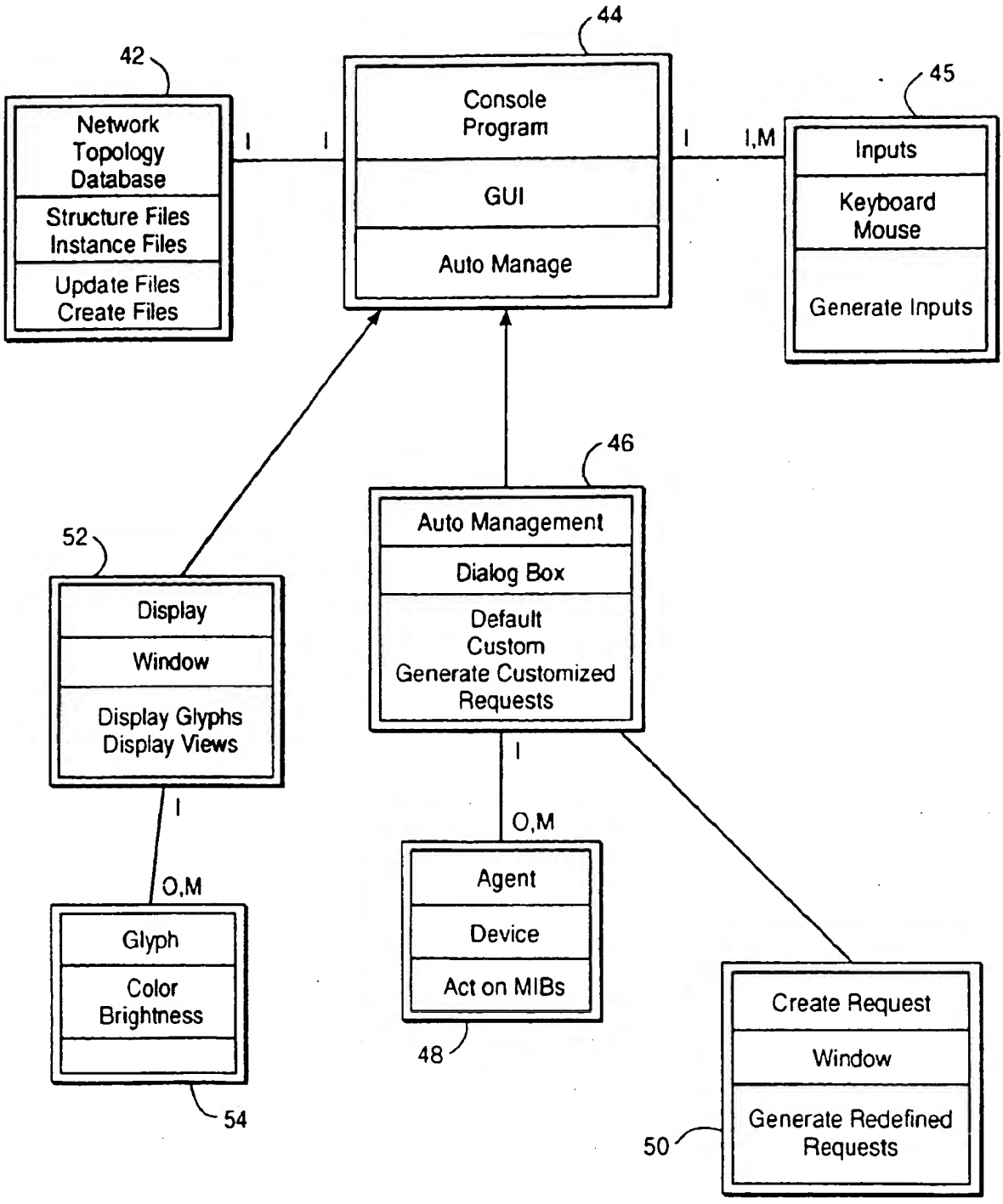
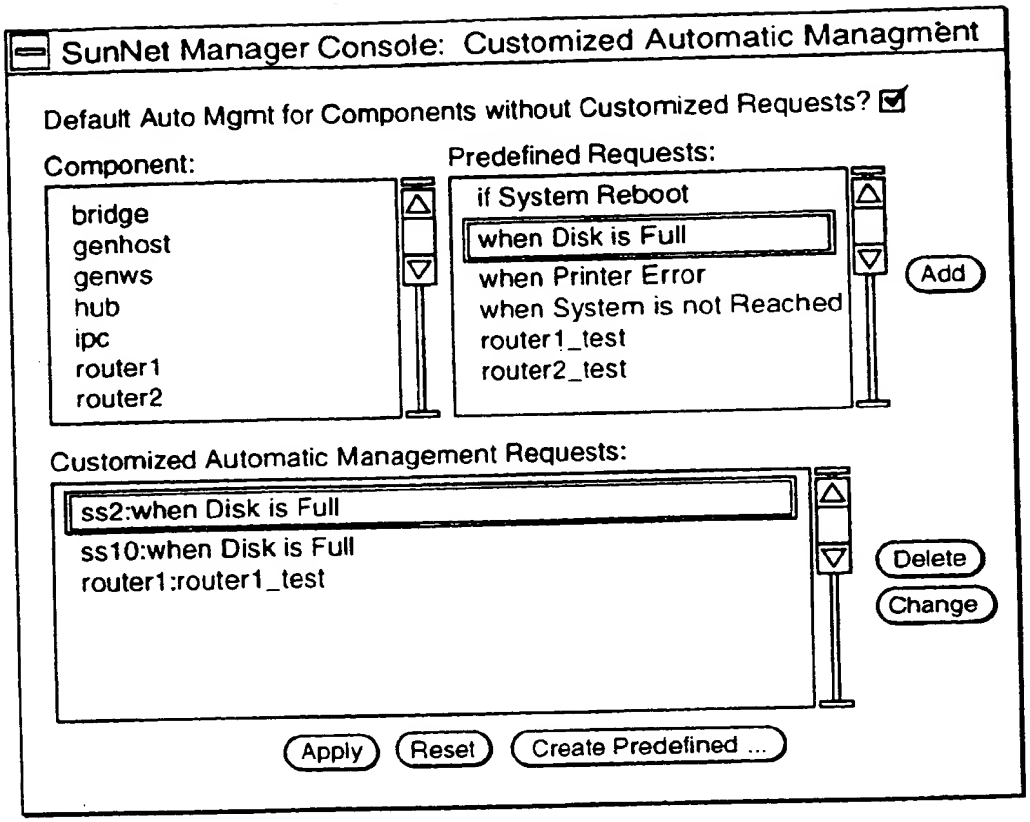


FIGURE 3

FIGURE 4



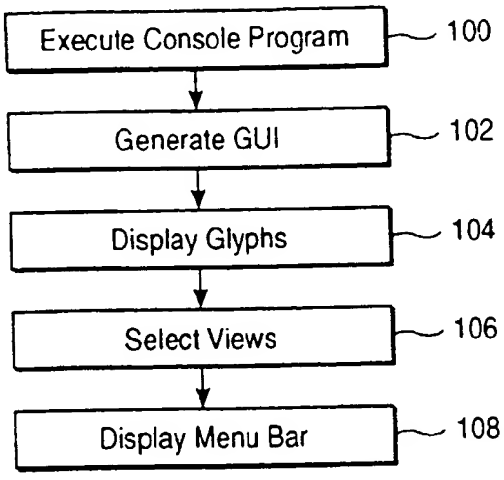


FIGURE 5

FIGURE 6

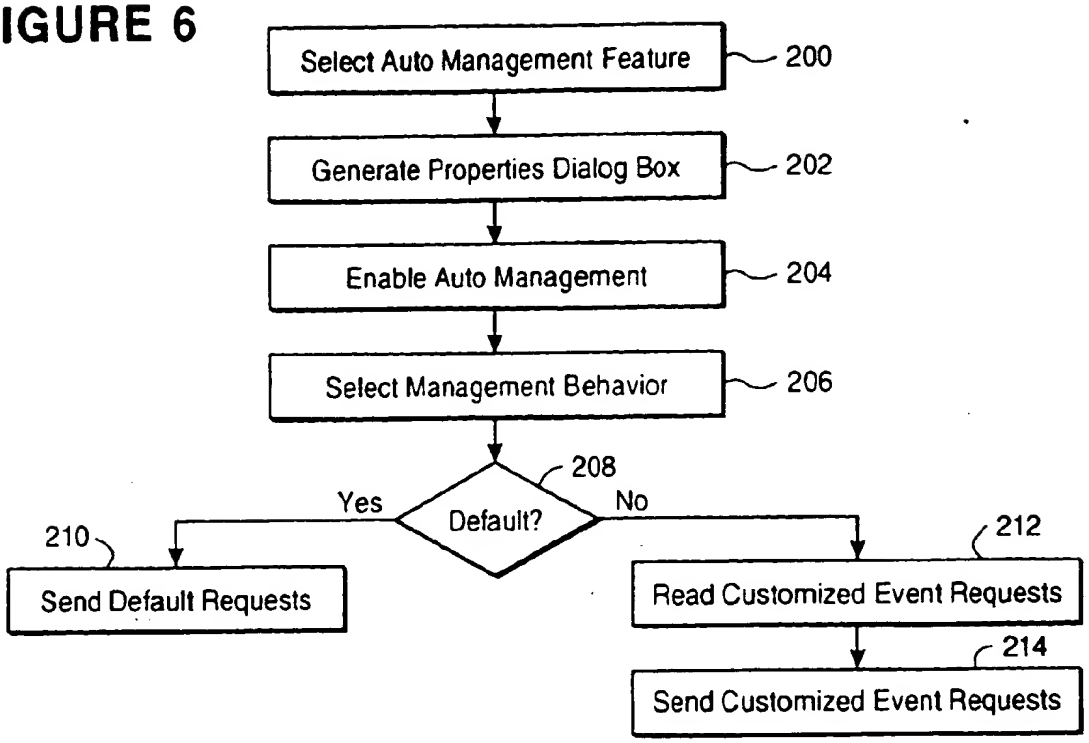


FIGURE 7

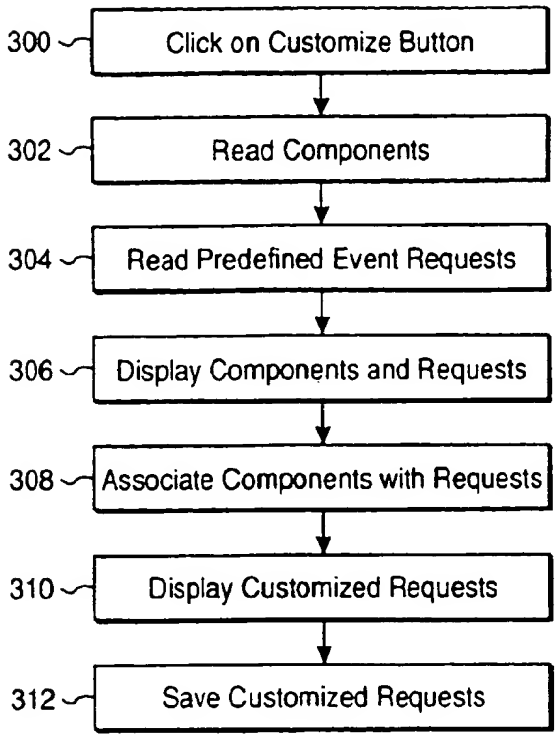


FIGURE 8